UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/929,147	08/14/2001	Andreas H. Kuehnel	2007.015900	8938

22879 7590 07/08/2004

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
TRUONG, CAM Y T	
ART UNIT	**PAPER NUMBER**
2172	

DATE MAILED: 07/08/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/929,147	KUEHNEL, ANDREAS H.	
	Examiner	Art Unit	
	Cam Y T Truong	2172	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 May 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-17,19-24,26-31,33-38,40-45,47-52 and 54-57* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) *1-8,10-13,15-17,19,20,22-24,26,27,29-31,33,34,36-38,40,41,43-45,47,48,50-52,54,55 and 57.* is/are rejected.

7)☐ Claim(s) *9,14,21,28,35,42,49 and 56* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant has amended claims 1, 10, 15, 22, 29, 36, 43, 50, 47, 54 in the amendment filed on 5/26/2004. Claims 1-17, 19-24, 26-31, 33-38, 40-45, 47-52, 54-57 are pending in this Office Action.

Applicant's arguments with respect to claims 1-17, 19-24, 26-31, 33-38, 40-45, 47-52, 54-57 has been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 3, 5, 7, 15-17, 20, 22-24, 27, 29-31, 34, 36-38, 41, 43-45, 48, 50-52, 55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Burrows (US 5966703).

As to claim 1, Macon teaches the claimed limitations:

"a plurality of clusters " as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47), "each cluster comprising a plurality of objects" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"and a second data structure indicating the state of the clusters" as each cluster has a corresponding entry in the FAT that describes its current use: available, reserved, assigned to a file or unusable. For example, 0x0000 signifies an available cluster and 0xFFFF signifies an end-of cluster chain. FAT is represented as a second data structure (col. 4, lines 39-42).

"a first data structure indicating a state of the objects" as the root directory is known as the files area, which may be viewed as pools of clusters. Each cluster contains one or more sectors. Boot sector indicates reserved sectors, starting at 0 (two bytes) is represented as the state of sector (fig. 2, col. 4, lines 34-40).

Macon does not explicitly teach the claimed limitation "a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure, wherein each word comprises a plurality of bits". Burrows teaches the size 253 can be expressed as the number of bytes of a page. The size information can help a user determine the amount of bandwidth needed to download the page. A byte contains words, which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Burrows's teaching of the size 253 can be expressed as the number of bytes of a page and a byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 3, Macon teaches the claimed limitation "a plurality of containers populated by the cluster and wherein at least some containers comprises files" as the root directory contains files (col. 3, lines 30-35).

As to claim 5, Macon teaches the claimed limitation "the objects comprise slots in the file" as each sector having a plurality of storage locations (col. 3, lines 18-19). As to claim 7, Macon teaches the claimed limitation "wherein at least one of the first and second data structures comprises a bitmap" as allocate bitmap for unit into temporary storage (figs. 3-4)

As to claim 7, Macon teaches the claimed limitation "wherein at least one of the first and second data structures comprises a bitmap" as allocate bitmap for unit into temporary storage (figs. 3-4).

As to claim 15, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS.

In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for

a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67).

Macon does not explicitly teach the claimed limitation "the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure, wherein each word comprises a plurality of bits associated with an implementation specific wordlength". Burrows teaches the size 253 can be expressed

as the number of bytes of a page. The size information can help a user determine the amount of bandwidth needed to download the page. A byte contains words, which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Burrows's teaching of the size 253 can be expressed as the number of bytes of a page and a byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claims 16, 23 and 30, Macon teaches the claimed limitations:

"constructing the first data structure" as FAT file system (fig. 6);

"constructing the second data structure" as directory (col. 4, lines 55-56).

As to claims 17, 24, and 31, Macon teaches the claimed limitation "wherein tracking the state for each of the plurality of objects in the first data structure or tracking the state for cluster of the memory like objects in the second data structure includes tracking a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be

understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claims 20, 27 and 34, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is

taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 22, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to

620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index

is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67).

Macon does not explicitly teach the claimed limitation "the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure; wherein each word comprises a plurality of electronic bits". Burrows teaches the size 253 can be expressed as the number of bytes of a page. The size information

can help a user determine the amount of bandwidth needed to download the page. A byte contains words, which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Burrows's teaching of the size 253 can be expressed as the number of bytes of a page and a byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 29, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF,

the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of

the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67).

Macon does not explicitly teach the claimed limitation "the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure; wherein the words each have a wordlength based on a maximum number of bits handled by a processor that executes an operating system". Burrows teaches the size 253 can be expressed as the number of bytes of a page. The size information can help a user determine the amount of bandwidth needed to download the page. A byte contains words, which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Burrows's teaching of the size 253 can be expressed as the number of bytes of a page and a byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 36, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value;

otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the allocation and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to

serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "consulting at least one usage counter that indicates how many sets of adjacent binary bits are set in words of the directory structure". Burrows teaches the size 253 can be expressed as the number of bytes of a page. The size information can help a user determine the amount of bandwidth needed to download the page. A byte contains words, which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Burrows's teaching of the size 253 can be expressed as the number of bytes of a page and a byte contains words which have many bits to

Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claims 37, 44 and 51, Macon teaches the claimed limitation:

"constructing the allocation data structure" as FAT file system (fig. 6);

"constructing the directory data structure" as directory (col. 4, lines 55-56).

As to claims 38, 45 and 52, Macon teaches the claimed limitation "tracking in a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of

the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claims 41, 48 and 55, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the slots" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-

47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 43, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken

to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the allocation and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF,

the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as

whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "consulting at least one usage counter that indicates how many sets of adjacent binary bits are set in words of the directory structure, wherein each word comprises a plurality of bits". Burrows teaches the size 253 can be expressed as the number of bytes of a page. The size information can help a user determine the amount of bandwidth needed to download the page. A byte contains words, which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Burrows's teaching of the size 253 can be expressed

as the number of bytes of a page and a byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 50, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken

to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the first and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address

at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "consulting at least one usage counter that indicates how many sets of adjacent bits are set in words of the directory structure, wherein each word comprises a plurality of bits".

whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too.

Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "consulting at least one usage counter that indicates how many sets of adjacent binary bits are set in words of the directory structure". Burrows teaches the size 253 can be expressed as the number of bytes of a page. The size information can help a user determine the amount of bandwidth needed to download the page. A byte contains words which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Burrows's teaching of the size 253 can be expressed as the number of bytes of a page and a byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

5. Claims 2 and 4 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of and further in view of Lehman (USP 5732402).

As to claim 2, Macon and Lawrence disclose the claimed limitation subject matter in claim 1, except the claimed limitation "a plurality of containers populated by the clusters and control data associated with the containers". However, Lehman teaches that management of the LOB data space, including allocation of space and

storage/retrieval of data, is controlled by allocation pages. Allocation pages are represented as a plurality of container (col. 5, lines 43-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman's teaching of management of the LOB data space, including allocation of space and storage/retrieval of data, is controlled by allocation pages to Macon and Lawrence's system in order to set flags in storage for indicating whether a space is currently occupied or is free to be used.

As to claim 4 Macon and Lawrence disclose the claimed limitation subject matter in claim 3, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case, examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon and Lawrence's system in order to store a large amount objects in the context of a paging memory.

6. Claim 6 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Burrows and further in view of Yamagami et al (or hereinafter "Yamagami") (USP 6256282).

As to claim 6, Macon and Lawrence disclose the claimed limitation subject matter in claim 1, except the claimed limitation "wherein each cluster comprises 16 objects". However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon, Lawrence's system in order to store a large data object in a cluster.

7. Claims 8, 19, 26, 33, 40, 47 and 54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Burrows and further in view of Zwilling et al (or hereinafter "Zwilling") and Orcutt (USP 6377958).

As to claims 8, 19, 26, 33, 40, 47 and 54, Macon and Burrows discloses the claimed limitation subject matter in claims 1, 10, except the claimed limitation "at least one other counter selected from the group consisting of: a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file. Orcutt

teaches count indicating the number of free clusters available for temporary use during a cluster remapping operation (col. 20, lines 55-56).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file and Orcutt's teaching of count indicating the number of free clusters available for temporary temporary to Macon, Lawrence and Burrows in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

8. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lawrence et al (or hereinafter "Lawrence") (USP 6253300) and Burrows.

As to claim 10, Macon teaches the claimed limitations:

"a plurality of files" as the root directory is the root of all files /subdirectories (col. 4, line 55-56);

"a plurality of clusters populating each file," as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47) "each cluster comprising a plurality of slots" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"a directory bitmap indicating the state of the clusters" as in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "an allocation bitmap indicating a state of the slots; a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure, wherein each word comprises a plurality of bit". Lawrence teaches bitmap indicates which sectors or clusters are being used. This information shows that the bitmap indicates a state of sectors as being used. This bitmap is represented as an allocation bitmap. Sectors are presented as

slots. Being used is represented as a state of slots (col. 12, lines 53-55). Burrows teaches the size 253 can be expressed as the number of bytes of a page. The size information can help a user determine the amount of bandwith needed to download the page. A byte contains words, which have many bits. The size 253 is represented as a counter indicative (col. 8, lines 47-50, fig. 6).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lawrence's teaching of bitmap indicating which sectors or clusters are being used and Burrows's teaching of the size 253 can be expressed as the number of bytes of a page and a byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

9. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lawrence et al (or hereinafter "Lawrence") (USP 6253300) and Burrows and further in view of Lehman (USP 5732402).

As to claim 11, Macon, Burrows and Lawrence disclose the claimed limitation subject matter in claim 3, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case,

examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon, Burrows and Lawrence's system in order to store a large amount objects in the context of a paging memory.

10. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lawrence et al (or hereinafter "Lawrence") (USP 6253300) and Burrows and further in view of Yamagami et al (or hereinafter "Yamagami") (USP 6256282).

As to claim 12, Macon, Lawrence and Burrows disclose the claimed limitation subject matter in claim 1, except the claimed limitation "wherein each cluster comprises 16 slots". However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon's system in order to store a large data object in a cluster.

11. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Lawrence and Burrows and further in view of Zwilling et al (or hereinafter "Zwilling") and Orcutt (USP 6377958).

As to claim 13, Macon, Lawrence and Burrows disclose the claimed limitation subject matter in claim 10, except the claimed limitation "at least one other counter selected from the group consisting of : a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file. Orcutt teaches count indicating the number of free clusters available for temporary use during a cluster remapping operation (col. 20, lines 55-56).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file and Orcutt's teaching of count indicating the number of free clusters available for temporary to Macon, Lawrence and Burrows in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

12. Claim 57 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Burrows and further in view of Bilbrey et al (or hereinafter "bilbrey") (USP 5227863).

As to claim 57, Macon and Burrows disclose the claimed limitation subject matter, except the claimed limitation "wherein the number of sets of adjacent bits is

selected from the group consisting of 2, 4, 8, 16, 32 and 64". Bilbrey teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits (col. 40, lines 60-67).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Bilbrey's teaching of teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits to Macon and Burrows's system in order to store or display data on a pixel by pixel basis.

13. Claims 1, 3, 5, 7, 15-17, 20, 22-24, 27, 29-31, 34, 36-38, 41, 43-45, 48, 50-52, 55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Millett et al (or hereinafter "Millett") (US 5717912).

As to claim 1, Macon teaches the claimed limitations:

"a plurality of clusters " as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47), "each cluster comprising a plurality of objects" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"and a second data structure indicating the state of the clusters" as each cluster has a corresponding entry in the FAT that describes its current use: available, reserved, assigned to a file or unusable. For example, 0x0000 signifies an available cluster and 0xFFFF signifies an end-of cluster chain. FAT is represented as a second data structure (col. 4, lines 39-42).

"a first data structure indicating a state of the objects" as the root directory is known as the files area, which may be viewed as pools of clusters. Each cluster contains one or more sectors. Boot sector indicates reserved sectors, starting at 0 (two bytes) is represented as the state of sector (fig. 2, col. 4, lines 34-40).

Macon does not explicitly teach the claimed limitation "a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure, wherein each word comprises a plurality of bits". Millett teaches the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words, which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Millett's teaching of the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 3, Macon teaches the claimed limitation "a plurality of containers populated by the cluster and wherein at least some containers comprises files" as the root directory contains files (col. 3, lines 30-35).

As to claim 5, Macon teaches the claimed limitation "the objects comprise slots in the file" as each sector having a plurality of storage locations (col. 3, lines 18-19).As to claim 7, Macon teaches the claimed limitation "wherein at least one of the first and

second data structures comprises a bitmap" as allocate bitmap for unit into temporary storage (figs. 3-4).

As to claim 7, Macon teaches the claimed limitation "wherein at least one of the first and second data structures comprises a bitmap" as allocate bitmap for unit into temporary storage (figs. 3-4).

As to claim 15, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free

(0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address

at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67).

Macon does not explicitly teach the claimed limitation "the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure, wherein each word comprises a plurality of bits associated with an implementation specific wordlength". Millett teaches the structure includes controls, this word's reference count in the index, and number of bytes for index piece. A byte contains words, which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Millett's teaching of the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claims 16, 23 and 30, Macon teaches the claimed limitations:

"constructing the first data structure" as FAT file system (fig. 6);

"constructing the second data structure" as directory (col. 4, lines 55-56).

As to claims 17, 24, and 31, Macon teaches the claimed limitation "wherein tracking the state for each of the plurality of objects in the first data structure or tracking the state for cluster of the memory like objects in the second data structure includes tracking a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625

examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claims 20, 27 and 34, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the objects" as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more

sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 22, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to

manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free

nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67).

Macon does not explicitly teach the claimed limitation "the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure; wherein each word comprises a plurality of electronic bits". Millett teaches the structure includes controls, this word's reference count in the index, and number of bytes for index piece. A byte contains words, which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Millett's teaching of the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 29, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a second data structure" in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47), "and consulting at least one of the first and second data structures to manage the objects" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented

as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20);

" tracking a state for each of a plurality of objects populating a container in a first data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should

track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as objects (col. 4, lines 37-40; col. 8, lines 25-47);

"consulting at least one usage counter to manage the objects" as (fig. 5, col. 7, lines 50-67).

Macon does not explicitly teach the claimed limitation "the at least one usage counter indicates how many sets of adjacent bits are set in words of the second data structure; wherein the words each have a wordlength based on a maximum number of bits handled by a processor that executes an operating system". Millett teaches the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words, which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Millett's teaching of the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 36, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process

begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the allocation and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for

clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "consulting at least one usage counter that indicates how many sets of adjacent binary bits are set in words of the directory structure Millett teaches the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words, which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Millett's teaching of the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claims 37, 44 and 51, Macon teaches the claimed limitation:

"constructing the allocation data structure" as FAT file system (fig. 6);

"constructing the directory data structure" as directory (col. 4, lines 55-56).

As to claims 38, 45 and 52, Macon teaches the claimed limitation "tracking in a bitmap" as a request to read a FAT storage unit can be replaced by an unpack function which converts the FAT storage unit information stored as packed records and the end-of-file bitmap into an unpacked form. Referring therefore now to FIG. 6, the explanation

will now proceed to the unpacking of the FAT. The coding of steps as described into instructions suitable to control the system processor will be understood to one having ordinary skill in the art of programming. The process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster

(0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47).

As to claims 41, 48 and 55, Macon teaches the claimed limitation "consulting at least one list containing information extracted from usage counters to manage the slots"

as the remainder of the volume after the root directory is known as the files area which may be viewed as pools of clusters, each containing one or more logical sectors. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 20-47; col. 4, lines 37-39). Since cluster includes one or more sectors, in case a cluster includes a sector; thus, when the system sets up a counter for cluster. It means that the system sets up a counter for sector and extracts value of count to manage the sectors.

As to claim 43, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit;

pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the allocation and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical

sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for

clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "consulting at least one usage counter that indicates how many sets of adjacent binary bits are set in words of the directory structure, wherein each word comprises a plurality of bits".

Millett teaches the structure includes controls, this word's reference count in the index, and number of bytes for index piece. A byte contains words, which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Millett's teaching of the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

As to claim 50, Macon teaches the claimed limitations:

"tracking a state for cluster of the memory like objects in a directory data structure" as the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record

Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47; col. 4, lines 35-40);

"and consulting at least one of the first and directory data structures to manage the slots" as the file allocation tables are followed by the volume files. The boot sector contains the number of sectors per fat. This information shows that the system consults more than one fat to manage the sectors. The first fat is represented as first data structure and the second fat is represented as second data structure (col. 4, lines 10-20).

"tracking a state for each of a plurality of slots populating a file in a allocation data structure" as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters, each containing one or more logical sectors. In fig. 6 shows the process begins at 610, where various parameters

are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one. Since clusters contain one or more logical sectors, thus, when the system tracks the state of clusters to determine them as being free or bad, the system should track the state of sectors of clusters too. Being free or bad is presented as a state for clusters or sectors. Sectors are represented as slots (col. 4, lines 37-40; col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "consulting at least one usage counter that indicates how many sets of adjacent bits are set in words of the directory structure, wherein each word comprises a plurality of bits". Millet teaches the

structure includes controls, this word's reference count in the index, and number of bytes for index piece. A byte contains words, which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Millett's teaching of the structure includes controls, this word's refernce count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

14. Claims 2 and 4 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Millett and further in view of Lehman (USP 5732402).

As to claim 2, Macon and Lawrence disclose the claimed limitation subject matter in claim 1, except the claimed limitation "a plurality of containers populated by the clusters and control data associated with the containers". However, Lehman teaches that management of the LOB data space, including allocation of space and storage/retrieval of data, is controlled by allocation pages. Allocation pages are represented as a plurality of container (col. 5, lines 43-45).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman's teaching of management of the LOB data space, including allocation of space and storage/retrieval of data, is controlled by

allocation pages to Macon and Lawrence's system in order to set flags in storage for indicating whether a space is currently occupied or is free to be used.

As to claim 4 Macon and Lawrence disclose the claimed limitation subject matter in claim 3, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case, examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon and Lawrence's system in order to store a large amount objects in the context of a paging memory.

15. Claim 6 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Millett and further in view of Yamagami et al (or hereinafter "Yamagami") (USP 6256282).

As to claim 6, Macon and Lawrence disclose the claimed limitation subject matter in claim 1, except the claimed limitation "wherein each cluster comprises 16 objects".

However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon, Lawrence's system in order to store a large data object in a cluster.

16. Claims 8, 19, 26, 33, 40, 47 and 54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Millett and further in view of Zwilling et al (or hereinafter "Zwilling") and Orcutt (USP 6377958).

As to claims 8, 19, 26, 33, 40, 47 and 54, Macon and Burrows discloses the claimed limitation subject matter in claims 1, 10, except the claimed limitation "at least one other counter selected from the group consisting of : a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file. Orcutt teaches count indicating the number of free clusters available for temporary use during a cluster remapping operation (col. 20, lines 55-56).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file and Orcutt's teaching of count indicating the number of free clusters available for temporary temporary to Macon, Lawrence and Burrows in order

to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

17. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lawrence et al (or hereinafter "Lawrence") (USP 6253300) and Millett.

As to claim 10, Macon teaches the claimed limitations:

"a plurality of files" as the root directory is the root of all files /subdirectories (col. 4, line 55-56);

"a plurality of clusters populating each file," as the remainder of the volume after the root directory is known as the files area, which may be viewed as pools of clusters. A file B uses cluster 6, 7 and 8. A file A uses clusters 3, 4 and 5 (col. 4, lines 34-36; col.6, lines 42-47) "each cluster comprising a plurality of slots" as each cluster contains one or more logical sectors as (col. 4, lines 38-39);

"a directory bitmap indicating the state of the clusters" as in fig. 6 shows the process begins at 610, where various parameters are initialized, including: MAXCLUSTERS being set to the number of clusters in a particular FAT storage unit; COUNT is set to zero; pCurrentRec is defined as a pointer to the first record in the FAT storage unit; pBitmap is defined as a pointer to a bitmap for a given FAT storage unit; Current Record Index is set to zero; and Current Bitmap Index is set to zero. At decision block 615, the value of COUNT is compared to MAXCLUSTERS. In the event

that COUNT is greater than MAXCLUSTERS, the YES branch is taken to 620 to exit the unpack function; otherwise, the NO branch is taken to serve as input to decision block 625. Decision block 625 examines whether the value of the bit at the position defined by the value of pBitmap+Current Bitmap Index is set. If set, the YES branch is taken where 630 assigns the value of the current cluster address at 0xFFFF, the end-of-file cluster value; otherwise the NO branch is taken to decision block 635. At decision block 635, the current cluster address is examined as constituting a free (0x0000) or bad cluster (0xFFF7). If the given cluster is neither free nor bad, the NO branch is taken to 640, where the Current Record Index is incremented by one (col. 8, lines 25-47).

Macon does not explicitly teach the claimed limitation "an allocation bitmap indicating a state of the slots; a counter indicative of a number of sets of adjacent bits that are set in words of the second data structure, wherein each word comprises a plurality of bit". Lawrence teaches bitmap indicates which sectors or clusters are being used. This information shows that the bitmap indicates a state of sectors as being used. This bitmap is represented as an allocation bitmap. Sectors are presented as slots. Being used is represented as a state of slots (col. 12, lines 53-55). Millett teaches the structure includes controls, this word's reference count in the index, number of bytes for index piece. A byte contains words which have many bits (col. 9, lines 40-55, fig. 3)

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lawrence's teaching of bitmap indicates which sectors or clusters are being used. This information shows that the bitmap indicates a state of

sectors as being used and Millett's teaching of the structure includes controls, this word's refernce count in the index, number of bytes for index piece. A byte contains words which have many bits to Macon's system in order to track or allocate data in memory correctly and reduce number of access operations necessary to store data.

18. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lawrence et al (or hereinafter "Lawrence") (USP 6253300) and Millett and further in view of Lehman (USP 5732402).

As to claim 11, Macon, Millett and Lawrence disclose the claimed limitation subject matter in claim 3, except the claimed limitation "wherein the file is a page file or a swap file". However, Lehman teaches that the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space. Applicant shows the file is a page file or a swap file. In this case, examiner indicates the file is a page file. Thus, the space allocation file of pages is represented as a page file (col. 5, lines 50-55).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Lehman teaching of the pages in the space allocation file include a means of indicating free blocks of storage location controls the storage of data in the buddy space to Macon, Burrows and Lawrence's system in order to store a large amount objects in the context of a paging memory.

19. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Lawrence et al (or hereinafter "Lawrence") (USP 6253300) and Millett and further in view of Yamagami et al (or hereinafter "Yamagami") (USP 6256282).

As to claim 12, Macon, Lawrence and Millett disclose the claimed limitation subject matter in claim 1, except the claimed limitation "wherein each cluster comprises 16 slots". However, Yamagami teaches that one cluster is constituted by 16 sectors (col. 16, line 9).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Yamagami's teaching of one cluster is constituted by 16 sectors to Macon's system in order to store a large data object in a cluster.

20. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon in view of Lawrence and Millett and further in view of Zwilling et al (or hereinafter "Zwilling") and Orcutt (USP 6377958).

As to claim 13, Macon, Lawrence and Millett disclose the claimed limitation subject matter in claim 10, except the claimed limitation "at least one other counter selected from the group consisting of : a counter of how many free pages a cluster has; and counter of how many free clusters are in the container". Zwilling teaches determining the number of used and free pages in the file. Orcutt teaches count

indicating the number of free clusters available for temporary use during a cluster remapping operation (col. 20, lines 55-56).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Zwilling's teaching of determining the number of used and free pages in the file and Orcutt's teaching of count indicating the number of free clusters available for temporary to Macon, Lawrence and Millett in order to indicate status of pages or clusters as being either available, reserved, assigned to a file for storing data.

21. Claim 57 is rejected under 35 U.S.C. 103(a) as being unpatentable over Macon, Jr. et al (or hereinafter "Macon") (USP 5715455) in view of Millett and further in view of Bilbrey et al (or hereinafter "bilbrey") (USP 5227863).

As to claim 57, Macon and Millett disclose the claimed limitation subject matter, except the claimed limitation "wherein the number of sets of adjacent bits is selected from the group consisting of 2, 4, 8, 16, 32 and 64". Bilbrey teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits (col. 40, lines 60-67).

It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to apply Bilbrey's teaching of teaches 16-bit bitmap, 64 bits, 32 bits, 8 bits, 4 bits, and 2 bits to Macon and Millett's system in order to store or display data on a pixel by pixel basis.

Allowable Subject Matter

22. Claims 9, 14, 21, 28, 35, 42, 49 and 56 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

As to claims 9 and 14, none of the available prior art of record teaches or fairly suggest "the second data structure contains clusters of at least four adjacent free bits; the second data structure is not empty, but contains no clusters of four adjacent free bits; the second data structure is empty, but allocation bitmap still shows free pages".

As to claims 21, 28, 35, 42, 49 and 56, none of the available prior art of record teaches or fairly suggest "a first list containing information indicating that the directory data structure for files in this list contains clusters of at least four adjacent free bits; a second list containing information indicating that the directory data structure for files in this list is not empty, but contains no clusters of four adjacent free bits; a third list containing information indicating that the directory data structure for files in this list is empty, but allocation bitmap still shows free slots".

Conclusion

22. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure

Bolan et al (US 6092071).

Contact Information

23. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cam Y T Truong whose telephone number is (703) 605-1169. The examiner can normally be reached on Monday to Firday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Breene, can be reached on (703) 305-9790. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Cam-Y Truong

7/2/04

SHAHID ALAM
PRIMARY EXAMINER